

DEĞİŞKENLER

▼ Değişkenler

▼ Değişkene neden ihtiyacımız var? Mantığı nedir?

Yazılımlar → Veriyi doğru bir şekilde işleyip çıktı veren yapılanmalardır.

- Amacımız: Yazılımda işlenecek veriyi yazılım adına RAM'e yerleştirebilmek için biz programcılar değişkenleri kullanabiliriz.
- Bir programcının işleyeceği veriyi ram de tutabilmesi için kullandığı bir yapılmadır.
- İşliyceğimiz veriyi ram de tutmalıyız. Ram kendi içerisinde bölmelere ayrılmıştır.
- ! Yazılım veriyi ram den alır, ram'e geri verir. Yazılım içerisinde veri tutmaz.
 - Bu işlenecek veriyi Ram'de tutabilmemiz için değişkenler kullanılır. Bu değişkenler Ram'den veriyi alır, yazılıma getirir. Yazılımdaki veriyi alır, ram'e verir.
 - Veritabanında falan da tutulur ama işleyemeyiz. İş yapıyosak %100 ram de çalışmalıyız.

▼ Value Type ve Primitive Type (Değer Türlü Değişkenler)

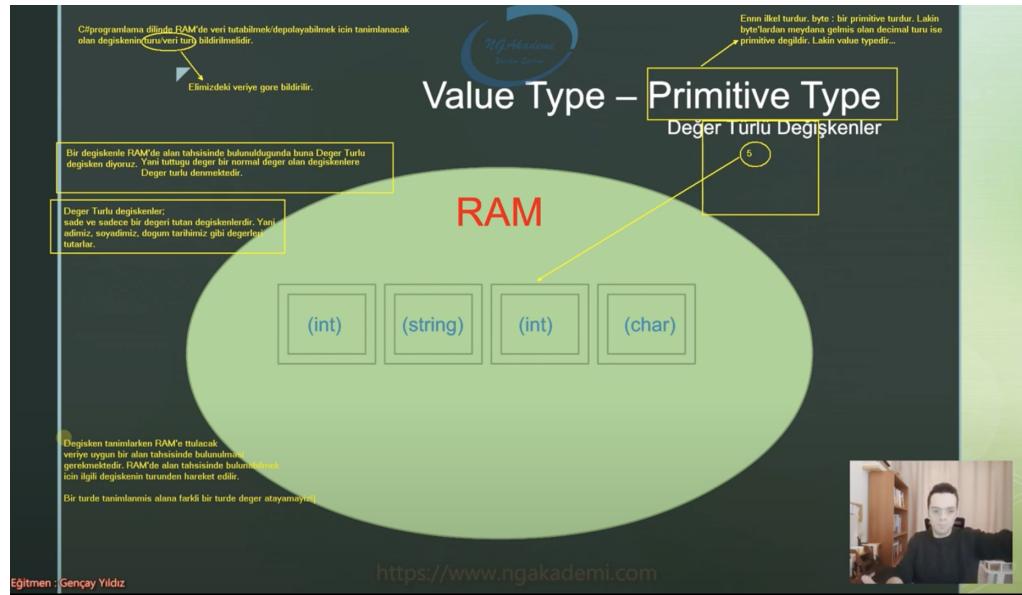
- Veriyi değişken vasıtasyyla RAM'de işleyeceğimizden, Ram'e yerleştirceğimiz verinin türünü baştan bildirmek zorundayız.
- Örneğin; int-int, string-string

▼ ValueType(değer türleri): Değişkenle Ram'de bir alan tahsisinde bulunduğuuzda değer türlü değişken diyoruz. Sade ve sadece bir değeri tutan değişkenlerdir. Örneğin benim adımı tutuyor.

Referans türlü değişkenler: Örneğin komple ben (Nesne) oop'de görücez

- Değişkenin türünden belirlenir ramdeki tür.
- Primitive Type: En ilkel türdür. Türetilmemiş veridir. Örneğin byte. Lakin byte'lardan meydana gelmiş decimal türü primitive değildir

çünkü türetilmiştir. Value type'dır.



Mantıksal	Tür	Açıklama ve Bellek Alanı	Max-Min Aralığı
Mantıksal	bool	Doğru veya Yanlış (1 Bit)	0-1 (True - False)
Metinsel	char	Karakterler (16 Bit)	16Bit Unicode
Sayısal	sbyte	İşaretli Tam Sayı (8 Bit)	-128 ile 127 arası
	byte	İşaretsiz Tam Sayı (8 Bit)	0 ile 255 arası
	short	İşaretli Tam Sayı (16 Bit)	-32.768 ile 32.767 arası
	ushort	İşaretsiz Tam Sayı (16 Bit)	0 ile 65.535 arası
	int	İşaretli Tam Sayı (32 Bit)	-2.147.483.648 ile 2.147.483.647 arası
	uint	İşaretsiz Tam Sayı (32 Bit)	0 ile 4.294.967.295 arası
	long	İşaretli Tam Sayı (64 Bit)	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arası
	ulong	İşaretsiz Tam Sayı (64 Bit)	0 ile 18.446.744.073.709.551.615 arası
Ondalıklı Sayılar	float	Tek Kayan Sayı (32 Bit)	$\pm 1,5 \times 10^{-45}$ ile $\pm 3,4 \times 10^{38}$ arası
	double	Çift Kayan Sayı (64 Bit)	$\pm 5 \times 10^{-324}$ ile $\pm 1,7 \times 10^{308}$ arası
	decimal	Ondalıklı Sayı (128 Bit)	$\pm 1,5 \times 10^{-28}$ ile $\pm 7,9 \times 10^{28}$ arası

Eğitmen: Gençay Yıldız

Metinsel ifadeleri tuttugumuz degiken turudur. Ahmed

string

- String bir referans türü değişkendir aslında. İleride öğreneceğiz.

▼ Primitive Type kontrolü nasıldır?

- Değer türlü değişkenlerde Primitive kontrolü nasıl yapılır?

```
Console.WriteLine(typeof(decimal).IsPrimitive);
Console.WriteLine(typeof(int).IsPrimitive); //hangi tür
Console.WriteLine(typeof(byte).IsPrimitive);
```

- Uygulama çalıştığında true false döndürür.

▼ Değişken türleri nelerdir

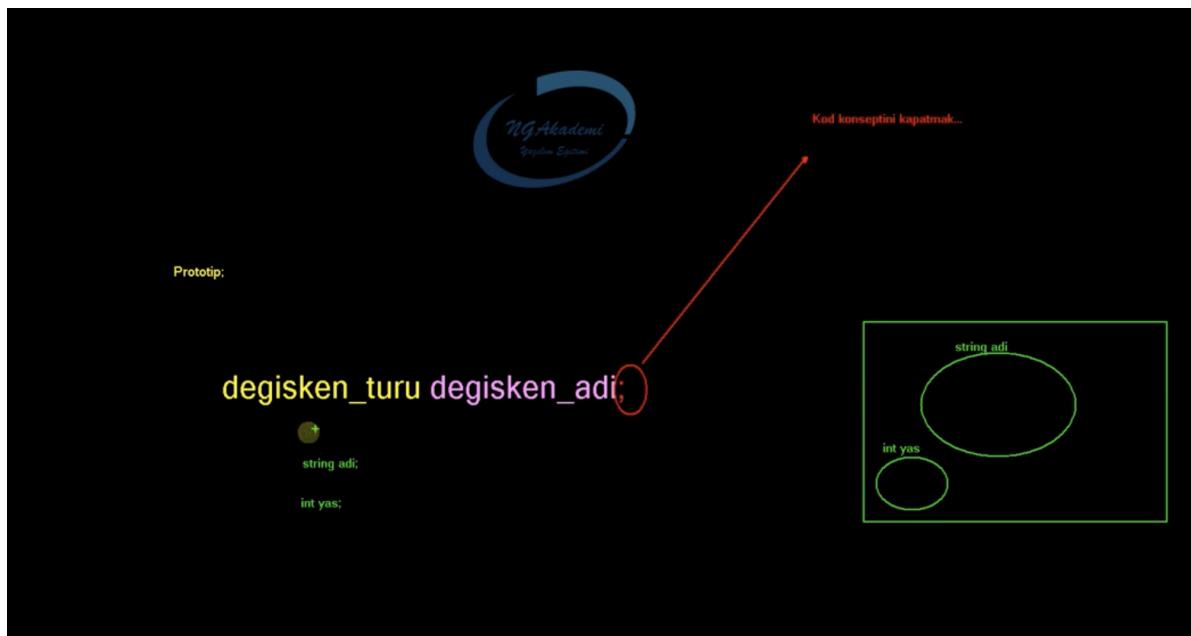
Tür	Açıklama ve Bellek Alanı	Max Min aralığı
bool	Doğru veya Yanlış (1 Bit)	0-1 [True/False]
char	Karakterler (16 Bit)	16Bit Unicode
sbyte	İpareti Tam Sayı (8 Bit)	-128 ile 127 arası
byte	İparetsiz Tam Sayı (8 Bit)	0 ile 255 arası
short	İpareti Tam Sayı (16 Bit)	-32.768 ile 32.767 arası
ushort	İparetsiz Tam Sayı (16 Bit)	0 ile 65.535 arası
int	İpareti Tam Sayı (32 Bit)	-2.147.483.648 ile 2.147.483.647 arası
uint	İparetsiz Tam Sayı (32 Bit)	0 ile 4.294.967.295 arası
long	İpareti Tam Sayı (64 Bit)	-9.223.372.036.854.775.808 ile 9.223.372.036.854.775.807 arası
ulong	İparetsiz Tam Sayı (64 Bit)	0 ile 18.446.744.073.709.551.615 arası
float	Tek Kayan Sayı (32 Bit)	$\pm 1.5 \times 10^{-4}$ ile $\pm 3.4 \times 10^0$ arası
double	Çift Kayan Sayı (64 Bit)	$\pm 5 \times 10^{-324}$ ile $\pm 1.7 \times 10^{308}$ arası
decimal	Ondalıklı Sayı (128 Bit)	$\pm 1.5 \times 10^{-32}$ ile $\pm 7.9 \times 10^{28}$ arası

{ Scope demek }

▼ C# dil özellikleri

- büyük küçük harf duyarlılığı vardır
- Tip güvenliği olan bir dildir. Tip güvenliği ise az önce bahsettiğimiz ram'e string ifade yazılması vs durumudur.
- JS ve Python'da tip güvenliği yoktur.

▼ Değişken tanımlama kuralları



```

1  using System;
2
3  namespace Degisenler
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              #region Değişken Tanımlama
10             int yas;
11             bool medenihal;
12             string soyadi;
13
14             RAM'in Yapısı(Stack)
15             Değişkenler RAM'de Nasıl Tutulur?
16             C# Kuralları
17             Değişken Tanımlama Kuralları
18             #endregion
19         }
20     }
21 }

```

Olusturma

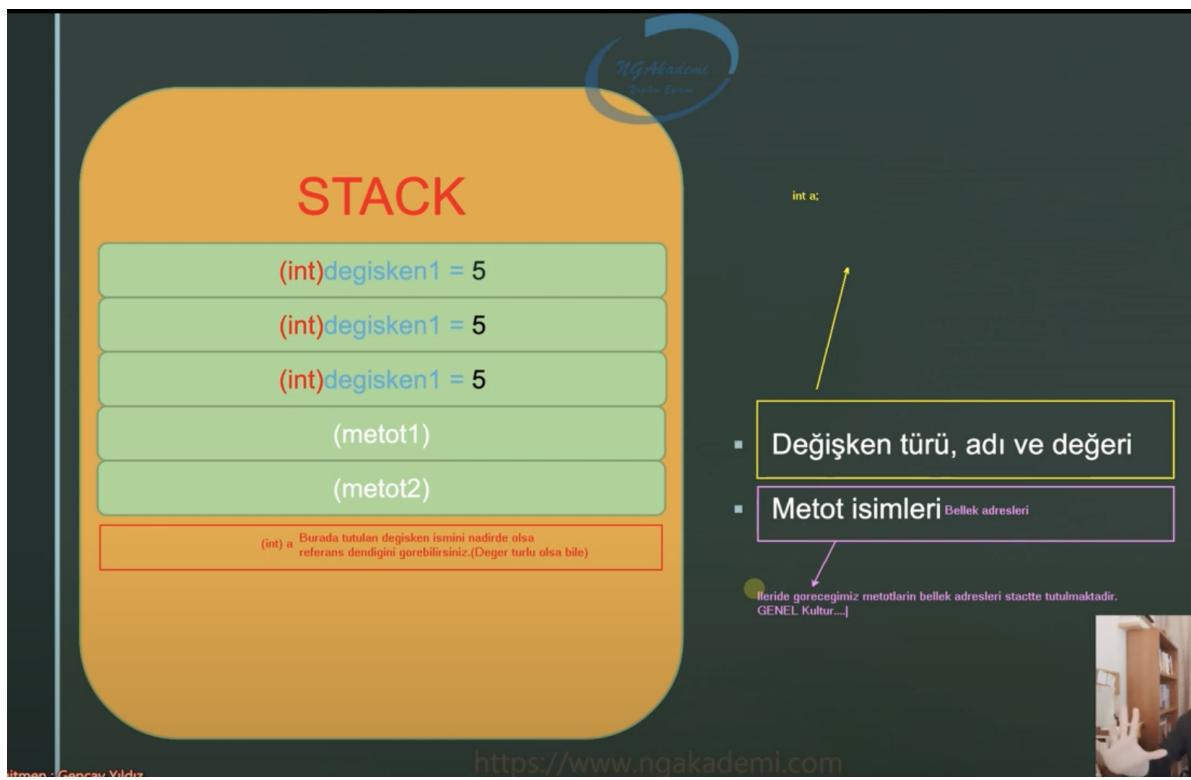
RAM	
(int) yas	
(bool) medenihal	
(string) soyadi	

▼ RAM'ın yapısı

STACK:İçerisinde değer türlü değişkenleri ve değerlerini tutar. x=5 gibi.

Değişken türü, adı ve değeri

Metot isimleri stack de tutulur.



```

 0 references
 static void Main(string[] args)
 {
 #region Değişken Tanımlama/Oluşturma
 //int yas;
 //bool medenihal;
 //string soyadi;

 [RAM'in Yapısı(Stack)]
 #region Değişkenler RAM'de Nasıl Tutulur?
 int yas;
 string adi;
 string soyadi;
 #endregion
 [Değişken Tanımlama Kuralları]
 [Değişken İsimlerini @ Operatörüyle Tanımlama]
 #endregion
 }

```

FIFO(First In First Out)

(int) yas
 (string) adi
 (string) soyadi

HEAP: Nesneleri tutabildiğimiz bölümdür.

▼ İsimlendirme Kuralları

Pascal case: Her kelimenin ilk harfi büyük yazılmalıdır.

Camel case: İlk kelime haricindeki tüm kelimelerin baş harfi büyük yazılmalıdır.

Snake Case: Tüm kelimeler küçük aralarında _ var.

▼ Değişken isimlerini @ operatörüyle tanımlama

```
string @x
```

//Değişken isimlerinde programatik keyword kullanılamaz. Eğer ki bir değişken isminde programatik olarak kullanılan bir keywordü vermek istersek bunu @ operatörü ile ezebiliriz. Örneğin;

```
string @static;
```

▼ Değişkene değer atama

- int x= 5;
- = atama operatördür. (Assign) Sağ tarafta verilen değeri soldaki değişkene atar, field'a, property vs. atar.
- Eğer ki bir değişken ismi assign operatörünün solunda çağrılıyorsa o alana değişkenin kendisi gelecektir. Sağında çağrıldığında farklı bir şey olacak ileride işlerez.
x(değişken) = value

- Dikkat edilmesi gerekenler:

1- Bir değişkene atanın en son değer geçerlidir.

int a = 5;

a=15;

a=20; ise

a 20 dir.

2-Tanımlanmış olan değişkene türüne uygun bir değer atılmalıdır.

▼ Değişkene değer atama kuralları

- Metinsel Değerler
 - **string:** Metinsel ifadeler çift tırnak içinde yazılmalıdır. "...."
 - Bir sayısal ifade metinsel olarak tutuluyorsa, yazılım açısından metinsel bir ifadedir.
 - Çift tırnak içerisinde yazılan tüm ifadeler string değişkenine atanıyor demektir.
"İrem". "234898998"
- Karaktersel Değerler
 - **char:** tek tırnak içerisinde yazılmalıdır. '....'
- Mantıksal Değerler
 - **bool:** True ya da False ile belirtilir.
 - true →1
false →0
- Sayısal Değerler
 - Değer atanırken direkt olarak değeri göndermeliyiz.
 - **Sayısal ifadelerde bir değer varsayılan olarak integer kabul edilir.**

▼ ONDALIKLI SAYILAR: Tüm ondalıklı sayılar tam sayıları karşılayabilir

- FLOAT: float türünde bir küsüratlı değer tutarken ilgili değerin sonuna f ya da F getirilmelidir.
 - 3.14f ya da 3.14F
- DOUBLE: double türünde bir küsüratlı değer tutarken ilgili değerin sonuna d ya da D getirilmelidir.
 - 3.14d ya da 3.14D
- DECİMAL: decimal türünde bir küsüratlı değer tutarken ilgili değerin sonuna m ya da M getirilmelidir.

- 3.14m ya da 3.14M
- **Ondalıklı sayılarda bir değer varsayılan olarak double kabul edilir.**

B

undan dolayı double da d ya da D girmek zorunda değilsin

▼ Literal Düzenlemler (C# 7.0)

Kompleks sayısal ifadeleri _ ile düzenlememizi sağlayan özellikleştir.

`int sayı = 1_000_000;` şeklinde yazabiliriz. okunabilirlik.

▼ Değişken türüne uygun default değer atama

Değişkenlerin defult değerleri: OOP yi öğrendikten sonra class içerisinde tanımlanan değişkenlerin default değerlerinin otomatik atandığını konuşacağız.
değersizlikle boş arasında fark vardır. Boş → boştur ama yine de bir karakterdir. Space dediğimiz.

`string → null ((değeri yok)`

`char → '0'`

`sayısal ifadelerin tümünde → 0 dır.`

`bool → false`

Default keywordü ile içerisinde verilen türü varsayılan değerini geri döndürür.

`bool x =default (bool);`

`int y = default(int);`

!!!! Main içerisinde oluşturulan değişkenlerin ilk değerleni manuel olarak atamaya özen gösteriniz.

Çünkü compiler kendisi veremez.

Açıklama: Class içerisinde tanımlanan değişkenlerde değer otomatik atanmaktadır. Lakin main içerisinde tanımlanan değişkenler de varsayılan değer atanmadığı için boş tanımlanan değerlerde ilk değeri manuel olarak vermediğimiz sürece işlem yapamayız.

```
int a;  
Console.WriteLine(a);  
int b=a;
```

Default Literals (7.1) ile

bool x = default; demem yeterlidir.

▼ Tanımlanmış değişkenin değerini okuma

- Bir değişkenin değerini elde edebilmek için değişkenin adından faydalanzırız.
- Bir değişkenin adın assign(=) operatörünün sağında yahut metodların parametrelerinde çağrılıyorsa ilgili değişkenin değeri gönderilir.

▼ Değişkenlerin faaliyet alanı (Scope)

C# da {} scope dur.

Bir scope içerisinde tanımlanmış değişkene o scope içerisinde her yerde erişilebilir. tanımlandıktan sonra.

Farklı scopelarda aynı isimde değişken olabilir.

▼ Global/local değişkenler

Bir değişken class scope'u içerisinde tanımlanıyorsa buna global değişken diyoruz.

Metotun içerisinde tanımlanıyorsa local'dır

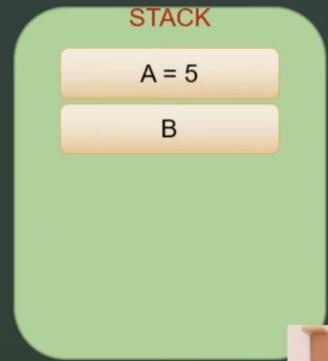
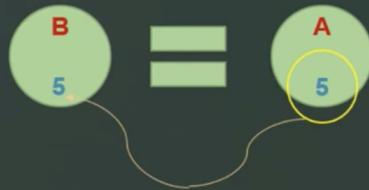
▼ Değişken arası değer atama.

Verisel açıdan iki davranış söz konusudur.

▼ Deep Copy

- Neticesinde eldeki veri çoğalır

Deep Copy



Deep copy(derin kopyalama) neticesinde eldeki veri
çoğalır/klonlanır.

<https://www.ngakademi.com>



```
ArasıDegerAtama
1  using System;
2
3  namespace DegiskenlerArasıDegerAtama
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              #region Deep Copy
10             //Derin koplama
11             //Eldeki veri çoğaltılar, klonlanır.
12
13             int a = 5;
14             int b = a;
15
16             a = a * 5;
17             Console.WriteLine(a);
18             Console.WriteLine(b);
19
20             #endregion
21             #region Shallow Copy
22
23             #endregion
24         }
25     }
26 }
```

Microsoft Visual Studio Debug Console

```
25
5
C:\Users\Gençay\Source\Repos\Application\DegiskenlerArasıDegerAtama\DegiskenlerArasıDegerAtama\DegiskenlerArasıDegerAtama\Program.cs (process 26172) exited with code 0.
To automatically close the console when debugging stops,
Press any key to close this window . . .
```

- Deep copy yapılmıştır ki, a'nın değişimi b'yi etkilememiştir.

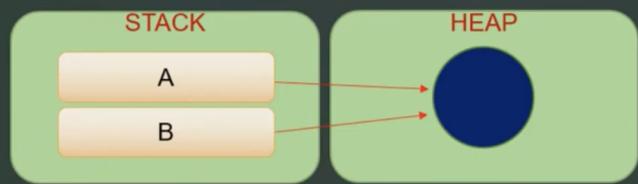
▼ Shallow Copy

Shallow Copy

Değişkenler arası değer atamalarında değeri türetmek/çoğaltmak/klonlamak yerine var olanı birden fazla referansla işaretlemeye dayalı kopyalama yöntemidir.

Bellekte birden fazla referansın tek bir veriyi işaret etmesidir.

Neticede ilgili değer bir değişikliğe uğradığında tüm işaretleyen referanslara bu değişiklik yansıyacaktır.



Normalde Değer Türü değişkenler default olarak Deep Copy edilirler.

Bu eğitimin ilerideki konusu olan `'ref keyword'` ile Değer Türü değişkenlerde nasıl Shallow Copy yapıldığını ele alacağız.

Shallow Copy OOP derslerinde ele alacağımız nesne ve referans arasındaki ilişkide vanayılan davranış olarak kabul edilmektedir. Bu konuya ilgili eğitimlerimizde devrindeki, nesneler üzerinde de ayrıyetten Deep Copy'nin nasıl gerçekleştirileceğini de ele alacağız...

